

PROGRAMMING
EXPERT

FAQ

Q I program using Visual Basic 6. Do I need to upgrade and, if so, to what?

A VB6 is dead. Even Microsoft's VB6 Resource Center pushes .NET more than the product it supposedly supports. You shouldn't start new projects in VB6. Ongoing projects should be ported to a new language if you want to keep developing them.

The natural upgrade path for VB6 programmers is VB 2005 (a .NET language). But VB 2005 is so different from VB6 that it may be as easy to learn another language. This month's bonus cover disc includes several .NET languages, which are especially suitable for the Windows programmer as .NET is more targeted at Windows than, say, Java. Visual C# 2005 is worth consideration, but Visual Basic 2005 is marginally easier to use.

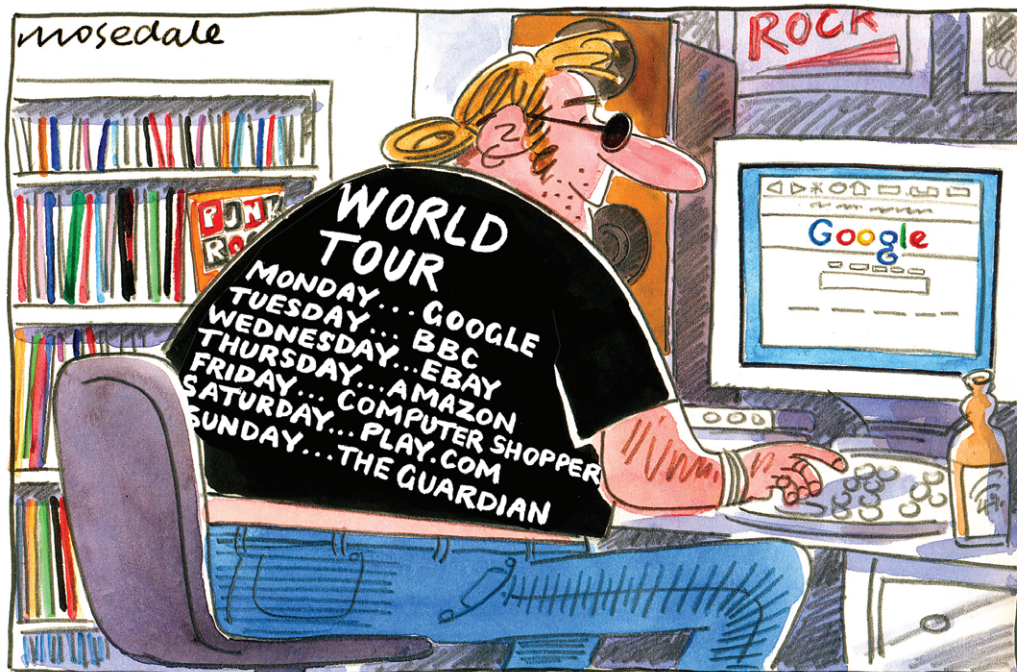
Mike James

IT author, lecturer
and programmer

mike@computershopper.co.uk

Home page changer

You can make your home page change each day if you tweak the setup. Mike James explains how to revitalise your browser



Have you ever wished that your browser's home page could be different every day? Perhaps there are web pages that you like to visit once a week. If so, a rotating home page would be just the ticket. Such an idea is fairly simple to implement, and the resulting system is robust and easy to use. You can program it using a simple language such as VBScript or JScript, and the only complication is that you must write a number of scripts and use several different Windows subsystems. It's a really good exercise in how to piece together a system from a number of small components.

In this project, we will learn how to add a custom menu item to Internet Explorer, work with the Registry, set up scheduled tasks using script, create and edit URL short cuts, run command-line utilities from script and write scripts that access the user's Favorites.

THE PLAN

Our aim is simple enough: change the browser's home page to one from a given list at a regular interval. There are various ways in which this idea could be elaborated upon. For example, you could assign each home page to a day of the week or set dates on which you want pages to open. In the spirit of keeping things simple, our home page utility will rotate through a list. To achieve this, we need to implement a way of adding a page to the list and some way of triggering the event that changes the home page.

A neat way to manage the list is to add a menu item to the right-click context menu in Internet Explorer. The extra menu item allows the user to add the current web page to the list of rotating home pages. Adding such a custom menu item to the context menu is a little-known technique, but it's the easiest way to extend or customise Explorer, and it's worth knowing.

If you want to know where to store the home page list, one way is to add each URL to a special folder in the Favorites folder. This has the advantage that the

rotating home pages just look like favourites and the user can add, delete or edit them in the usual way. The only disadvantage is that we have to find a way to manipulate the Favorites folder in a script.

Once the list of URLs is built, an easy way to run a program on a regular basis is to use the built-in Windows scheduler. The only problem is that its programmable interface is horrible. However, there are command-line utilities we can use to set up the schedule. And as with the Favorites, if we use a standard Windows facility to implement our idea, the user can then modify what happens using the standard tools. In this case, we'll use the scheduler wizard.

SETTING UP

Now we have the broad outline, it's time to get to work. As everything is going to be programmed in VBScript, which is easy to convert to JScript if you want to, we don't need any language environments installed. It is, however, worth making sure you have the latest version of the Windows Script Host and script engines installed; search for wsh at www.microsoft.com/downloads to check. You can create and modify all the scripts using nothing but Notepad, or you can use an editor such as Visual Web Developer Express Edition, which is included on this month's bonus cover disc.

As the whole system is composed of a number of scripts, we need an installation script that will set up everything and an uninstall script to return it all to the way it was. As the setup script does a great deal of the hard work, this is a good place to start.

Create a folder in which you are going to store all the scripts relating to this project. It doesn't matter what it's called or where it is. A good setup program should be able to install everything irrespective of the location of the project resources. Create a file called Setup.VBS in the directory.

It's a good idea to give the user a chance to change their mind before installing:



```
button=Msgbox("You are about to set
up Home Page Rotator - click OK to
continue or Cancel to abort",1)
if button = 1 then
    Setup
    MsgBox "Setup complete"
else
    msgbox "Setup cancelled - run
    setup again to install or refresh
    an installation"
end if
```

Now we have to create the Setup subroutine that does the actual installation. The subroutine has three parts: create folders and copy across scripts, create a new IE context menu item, and set up the schedule. To do all this, we need two auxiliary objects. The WshShell object, which provides ways of interacting with the system, and the FSO object, which provides ways of interacting with the file system:

```
Sub Setup
'create folders and copy files
Set WshShell = _
WScript.CreateObject("WScript.
Shell")
set fso=CreateObject( _
"Scripting.FileSystemObject")
```

We need to find out where the user's Program Files folder and System32 folder are. It would be simpler just to assume their standard names, but it isn't much extra work to ask the system for them just in case they are non-standard. The key to this is to read the environment variables:

```
'Get folder names
Set WshSysEnv = _
WshShell.Environment("PROCESS")
ProgFolder=WshSysEnv("ProgramFiles")
ProgFolder =ProgFolder & "\
HomePageRotator"
```

This gets the Program Files folder's pathname, usually C:\Program Files, and adds the name of the directory we are going to use to store all the application's files. Originally, we planned to store the entire project in this folder, but it turned out to be much easier to put the scheduled script into the System32 folder to avoid having to specify a path. The full pathname of the System32 folder can be found using:

```
Set WshSysEnv = _
WshShell.Environment("PROCESS")
Sys32=WshSysEnv("SystemRoot") &
"\System32"
```

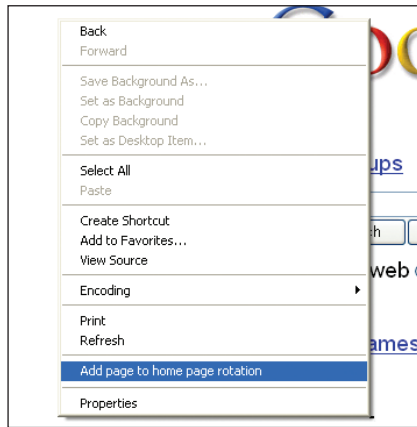
DIRECTORY ENQUIRIES

Finally, we need the current directory where all the project files to be installed, as well as the setup script, are stored:

```
currentdir= WshShell.current
directory
```

If the installation folder doesn't exist, we need to create it:

```
if not(fso.folderexists(ProgFolder))
then
    fso.createfolder(ProgFolder)
end if
```



▲ Adding a custom item to Internet Explorer's right-click menu is easy

Now we can copy the two key project files starting with the script that is triggered when the user selects the custom item in Explorer's context menu:

```
fso.copyfile currentdir & _
"\HR.htm",ProgFolder & "\",true
```

This script has to be stored in an HTML file. Exactly what it does will be explained shortly, but essentially it creates the list of home pages that we want to rotate.

The second script is the one that looks at the list and changes the home page to the next entry each time it is triggered:

```
fso.copyfile currentdir & _
"\rotate.vbs",Sys32 & "\",true
```

This script, rotate.vbs, is a standard VBScript and, again, we will return to how it works shortly.

ADDING A CUSTOM MENU ITEM

The second section of the setup program is very short. It adds the custom menu item to Internet Explorer and all you have to do is modify the Registry. As it is possible to break your Windows installation with a misjudged Registry edit, you should make sure that you create a restore point using System Restore before proceeding. This will allow you to install a working copy of the Registry if your program destroys it.

The Registry key that controls the extra menu items is HKey_Current_User\Software\Microsoft\Internet Explorer\MenuExt. All you have to do is create a subkey with the name of the menu item you want to appear and a value equal to the path of the HTML file that contains

the script that will be run when the user selects the item. There are two optional keys you can create to control exactly how your menu works. The Contexts subkey can be used to control when the menu item appears, such as only when the user right-clicks a picture. The Flags subkey can be used to give the script that is run a dialog box-style window. By default, the script runs without any window and appears whatever the user right-clicks. This is exactly what we need, so we can ignore the Contexts and Flags subkeys in this project.

Creating a key in the Registry is very easy if you use the RegWrite method of the WshShell object:

```
ProgURL="file://" & ProgFolder &
"\HR.htm"
MenuItem="Add page to home page
rotation"
WshShell.RegWrite _
"HKCU\Software\Microsoft\Internet
Explorer\MenuExt\" _
& MenuItem & "\",_
ProgURL, "REG_SZ"
```

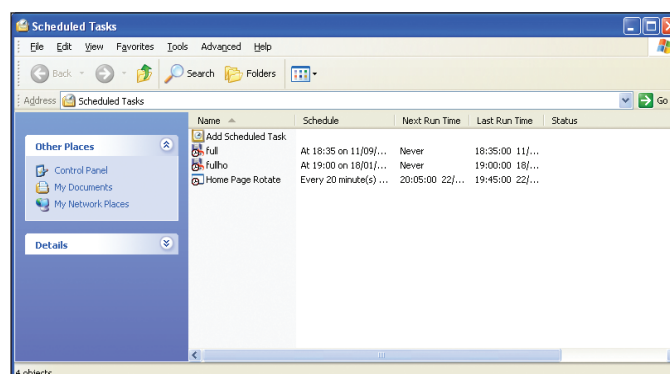
This adds the 'Add page to home page rotation' menu item as the key and sets its value to be the URL of the HR.htm web page, which contains the script to be run. After this line does its work, Internet Explorer has a new context menu item. However, we still need to create HR.htm.

WINDOWS SCHEDULER

The third and final part of the setup is to enter an event into the Windows scheduler. This doesn't sound difficult, but in practice it proved to be the hardest thing to get exactly right. There is no easy-to-use scripting interface to the scheduler; this is going to be put right in Vista, the next version of Windows. Under Windows XP, we can use the Windows Management Interface (WMI) or the command-line utility SchTasks. The WMI route is difficult and produces tasks that a user can't later edit using the wizard. Using a command-line utility is never very satisfactory, but in this case it is probably the best choice. You can look up SchTasks in Windows Help to get full details of how it works.

We need the user's username and password to be able to schedule a task. The username can be obtained automatically, but we have no choice but to prompt for the password:

```
Set Net = CreateObject("WScript.
Network")
User = Net.UserName
pass=InputBox( "Enter your
```



◀ The home page rotator is a Scheduled Task that can be edited or run in the usual way

HUNGRY MINDS RFID Toys

★★★★★

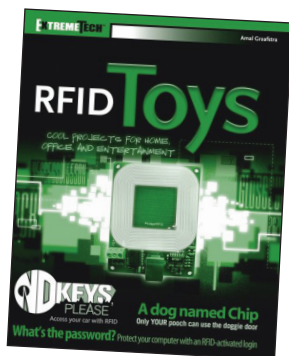
£11

If you enjoy building gadgets that combine software and hardware, you'll think this book is simply marvellous. Radio Frequency ID (RFID) tags are technology that's poised to change the way we shop, deliver things and keep track of details. Think of RFID as a barcode that you can read remotely using radio signals. This simple idea has many applications.

In this book, author Amal Graafstra takes us beyond the obvious with some imaginative projects. You can create an RFID key to log on to your PC, a pet tag that operates a pet door automatically, a key that unlocks your car or safe and so on. If you are as keen on the technology as the author, you can even have an RFID tag implanted under your skin. You'll find full details of this in the book.

An important feature of RFID is that the tags are cheap. This means you can use the tags to track lots of items such as books, CDs and even people. The hardware is fairly cheap, though currently hard to get hold of, and the explanations are easy enough to understand even if your electronics knowledge is minimal.

RFID is a growing area of interest, and this book promises a lot of good fun and perhaps even profit. If you have any interest in DIY hardware projects, this is essential reading.



SUMMARY

- ✓ A fun project book
- ✗ Hardware difficult to buy in UK

SUPPLIER www.amazon.co.uk
ISBN 0471771961
PAGES 336

APRESS

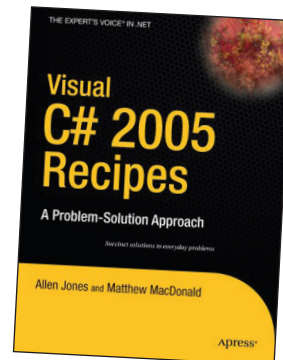
Visual C# 2005 Recipes: A Problem-Solution Approach

★★★★★

£22

Allen Jones and Matthew MacDonald have produced a collection of 'recipes' that tackle problems in a standard format. First they state the challenge, then outline the answer, present a detailed solution and finally make a few comments. The range of topics covered is impressive, from basic command-line projects through security, threads, XML, forms, graphics and databases to networking, cryptography, unmanaged code, patterns and so on. Some of the problems are clearly contrived so that the authors can explain some interesting idea or technique, but this is forgivable.

The only serious drawback of this book is that if you don't understand what it is that the recipes set out to solve, you are going to need another book to explain the basics to you. This is a book for at least intermediate C# programmers, and even experts are likely to be lost occasionally. But as long as you don't expect to cook every recipe, you are bound to enjoy this book.



SUMMARY

- ✓ Lots of good code to read
- ✗ Problems not fully explained

SUPPLIER www.amazon.co.uk
ISBN 1590595890
PAGES 600

```
password", _e
"Password")e
```

Before we create the new task, we need to make sure it isn't already in the list by attempting to delete it. The task is called Home Page Rotate and this is enough to identify and delete it:

```
cmdline ="schtasks /delete /tn
""Home Page Rotate"" /f"e
set sched=wshShell.exec(cmdline)e
do while sched.status=0e
wscript.sleep 100e
loop
```

The /f switch suppresses output from the command line so it's less messy when we use the WshShell's Exec method to run SchTasks. The status method returns 0 while the utility is running, so we loop until it is finished.

TASK CREATION

Now we can safely create the task, knowing that it can't already exist:

```
cmdline="schtasks /create /tn
""Home Page Rotate""e
cmdline=cmdline & " /tr rotate.vbs "e
cmdline=cmdline & " /sc minute /mo
20 /ru " & chr(34) & User & chr(34)
& " /rp " & chr(34) & pass & chr(34)e
e
set sched=wshShell.exec(cmdline)e
```

This uses the same SchTasks command as the delete operation, but now we have to specify the program to run when the task is triggered (rotate.vbs) and the schedule, every 20 minutes

in this case. You can set any time period you like, or you might want to add an InputBox that allows the user to specify the interval during setup. The double and triple quotes in the first line aren't a mistake; to include a double quote symbol in a string, you must use two double quotes. We have to put double quotes around the username and password to the command line because they may contain spaces. On this line, we've used the alternative technique of specifying the double-quote character by ASCII code using the chr(34) function.

Another complication is that the utility sends output to the two standard text streams. So after waiting a few moments, we need to read the StdOut and StdErr streams to see if there is anything to report to the user:

```
wscript.sleep 100e
input1 = sched.StdErr.ReadAlle
input2 = sched.StdOut.ReadAll
```

As long as everything has worked, input1 should be empty and input2 should contain an "everything is OK" style message. We can show the user both or just ignore them:

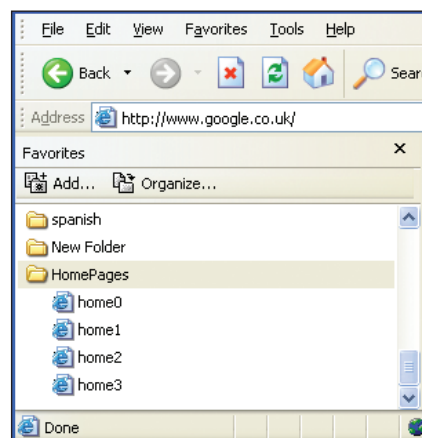
```
if input2<>"" thene
msgbox input2e
end ife
if input1<>"" Thene
msgbox input1e
end ife
end sub
```

That's all there is to the setup script. Now we have to write the two scripts that the setup script installs: one that adds pages to the rotated list and one that changes the home page.

CONTEXT MENU HANDLER

The HR.htm script is run when the user selects our new item from Internet Explorer's context menu. The main problem for the script is that it needs to interact with the original web page that the click event came from. This problem is easily solved once you know that the script runs in a hidden web page, and that web page has a full Document Object Model (DOM). This includes a window.external object, which has a property menuArguments that returns the window object of the web page that triggered the event. You can use that window object to access the HTML and other characteristics of the original page. In this case, all we need to obtain is the URL of the original page, which is easy.

Create a new file called HR.htm in the same directory as the setup script. Start with a comment describing what the script does:



▲ The list of rotating home pages is just a folder in Favorites



```
<!--
Script for adding current
page to home pages directory
-->
```

To retrieve the URL of the page that caused the script to run, all we need is:

```
<script language="VBScript">
'Get URL of calling page
set Win = window.external.
menuArguments
set Doc = Win.document
homeurl=doc.url
```

The window of the script is in this code window, which is hidden, but Win is the window of the original webpage.

FAVOURITE THINGS

Now we have the URL of the page, we need to add it to the user's Favorites. However, we're going to put our pages in a subdirectory called HomePages. First, we need to check this exists, and if it doesn't we'll have to create it. To do this and subsequent tasks, we use the same two objects that were so useful in setup.vbs:

```
set fso=CreateObject( _
"Scripting.FileSystemObject")
Set WshShell = CreateObject( _
"WScript.Shell")
```

The SpecialFolders method of the WshShell object can be used to find the path to any of the user's special directories, such as My Documents and Favorites:

```
UserFavs=WshShell.SpecialFolders( _
"Favorites")
UserFavs=UserFavs & "\\HomePages"
```

With the directory path set up for HomePages in the user's Favorites, we can check for, and if necessary create, the directory:

```
if not(fso.folderexists(UserFavs))
then
fso.createfolder(UserFavs)
end if
```

To save the URL in the HomePages folder, we need to create a URL short cut. Each short cut added to the folder is given a name in the sequence Home0, Home1, Home2 and so on. To name each one correctly, we simply count how many short cut files are already in the directory:

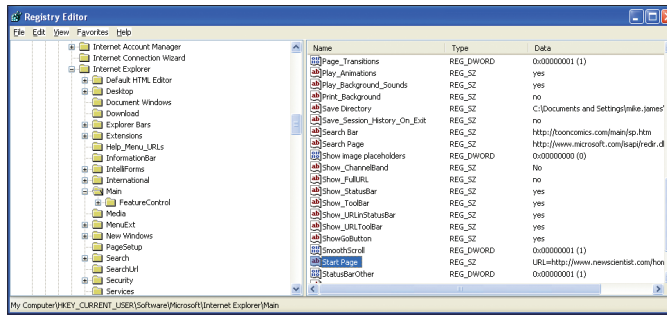
```
N=fso.GetFolder(UserFavs).Files.
Count
```

Creating a URL short cut is a two-step process. First we create the short cut object:

```
Set URLLink=WshShell.
CreateShortcut( _
UserFavs & "\\home" & N & ".url")
```

When you create the short cut, you specify the path and filename. Because this ends with .url, you get a URL short cut rather than a program short cut. Then you assign a target to the short cut and finally save it to disk:

```
URLLink.TargetPath=homeurl
```



◀ The Registry contains many Internet Explorer settings, including the current home page

```
URLLink.Save
</script>
```

This is the complete right-click menu handler, and it's compact considering what it does. For our project, we used only the document.url property of the original web page; other context menu operations could use its entire DOM.

THE ROTATOR

The final part of the jigsaw is the rotate.vbs script that looks in the user's Favorites and sets the home page to the next entry. This script is triggered by the Scheduler at regular intervals to rotate the home page periodically.

Create a new file called rotate.vbs in the same directory as setup.vbs. This starts by creating the FSO and WshShell objects:

```
set fso=CreateObject(_
"Scripting.FileSystemObject")
Set WshShell = WScript.
CreateObject( _
"WScript.Shell")
```

Before we can rotate the home page, we need to discover what it is, and this is stored in the Registry under the key HKey_Current_User\Software\Microsoft\Internet Explorer\Main\Start Page. To discover the current home page, you simply read the value associated with the key. To set a different home page, write its URL to the key's value. First retrieve the home page:

```
HomePage=wshShell.regread( _
"HKCU\Software\Microsoft\Internet
Explorer\Main\Start Page")
```

Next get the path to the HomePages folder in the user's Favorites as before:

```
UserFavs=WshShell.SpecialFolders( _
"Favorites")
UserFavs=UserFavs & "\\HomePages"
```

If this directory doesn't exist, the user hasn't set up rotating home pages and there's nothing to do. If it does exist, open and process the files in it:

```
if fso.folderexists(UserFavs) then
set Fs=fso.GetFolder(UserFavs).
Files
```

Unfortunately, even if it does exist, there might not be any files in it yet so we need to add:

```
if Fs.count>0 then
```

The simplest way to process the home pages is to read the short cuts into an array. Sadly, VBScript provides no way of accessing the URLs

from existing short cuts. Short cut files are easy to read manually as the URL is stored as plain text on the second line of the file. So to read the URLs we just open each short cut as a text file:

```
redim URLs(Fs.count)
i=0
for each f in Fs
set ts=f.OpenAsTextStream(1)
ts.readline
temp=ts.Readline
temp=Split(temp,"=")
URLS(i)=temp(1)
ts.close
if URLS(i)=HomePage then
found=(i+1) Mod FS.count
end if
i=i+1
next
```

Each file is opened in turn and the first line read and discarded. The second line has the format URL= followed by the actual URL, so we split temp on the '=' sign and take the second item, the URL, and store this in URLS(i).

After closing the file, we check to see if the current home page is one of the URLs in the list. If it is, the home page should be changed to the next URL in the list, (i+1). If this is the last item in the list, i+1 is beyond the end of the list. To stop this happening, we use the Mod operator, which makes found return to the start of the list when it reaches the end.

If the current home page isn't in the list, i is 0 by default, so no matter what happens, at the end of this process we have a valid value for i indicating the next URL in the list to use. All that remains is to set the Registry key to this URL:

```
NextURL=URLS(found)
wshShell.regWrite "HKCU\
Software\Microsoft\Internet Explorer\
Main\Start Page",NextURL, "REG_SZ"
end if
end if
```

TRYING IT OUT

Now we have all the scripts we need. If you run the setup script, the files will be copied to the correct locations and the Registry updated. After this, you will see a new right-click context menu item. You can use this to add the current web page to the HomePages folder of Favorites, and the Scheduler will start to cycle through these at the interval you chose. If you can't wait, go to the Scheduled Tasks folder, right-click on the Rotate home page task and select Run.

There are plenty of ways you can improve this project. An uninstall script is a good idea and quite easy to write. There is one on this month's cover disc, along with the other scripts. **CS**